Numerical Analysis (4/7)
Non-linear (systems of) equations

University of Luxembourg - 2024

Master in Mathematics

Hadrien Beriot

Outline

1. Introduction
2. Standard algorithms (1D)
3. Systems of non-linear equations
4. Summary

1. Introduction

Introduction

- Most relationships in nature are nonlinear in that **effects are not in direct proportion to their causes**
  - **Air resistance:** is proportional to the square of velocity (proportionality constant C_d = drag)

$$F_{drag} = C_d * v^2.$$

  - **Gravitational Force**: inversely proportional to the square of the distance between two masses, according to Newton's law of gravitation

$$F = Gm_1m_2/r^2$$

  - **Population Growth**: In ecosystems, population growth often follows a logistic model

$$\frac{dP}{dt} = r\,P\left(1 - \frac{P}{K}\right)$$

  Where r is the growth rate and K is the carrying capacity

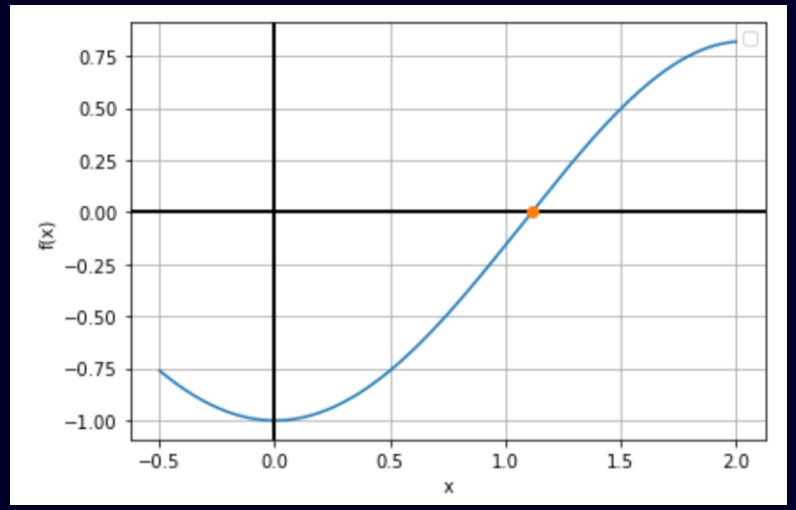
- Often, the relationships become more nonlinear as more physical features are included.

Non linear systems

- By analogy with a general system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$,we could write it as $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{y}$
- Answers “For what value of $\boldsymbol{x}$ does the nonlinear function $\boldsymbol{f}(\boldsymbol{x})$ take on the value **y**?”
- However, it is often presented under the form

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}.$$

- In 1D, this simply means that we seek the intersection of the curve defined by f with the x axis rather than the horizontal line $y =$ constant.



Root finding

- Thus, a general system of m nonlinear equations in n unknowns has the form

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$$

where $f: \mathbb{R}_n \to \mathbb{R}_m$, and we seek an n-vector x such that all m component functions of $\boldsymbol{f}(\boldsymbol{x})$ are zero **simultaneously**.

  - $m > n$: **overdetermined** - no solution in this strict sense → approximation in least squares sense
  - $m < n$: **underdetermined** - infinitely many solutions → optimization problem
  - $m = n$: scope of this course → as many equations as unknowns
- A solution value 0 such that $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$ is called a root of the equation, and a zero of the function $\boldsymbol{f}$.
- Thus, the problem is referred to as **zero finding** of **root finding**

Root finding

- An example of a nonlinear equation in one dimension is

$$f(x) = x^2 - 4\sin(x) = 0,$$

- for which one approximate solution is $x = 1.93375$. An example of a system of two nonlinear equations in two unknowns is

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} x_1^2 - x_2 + 0.25 \\ -x_1 + x_2^2 + 0.25 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- for which the solution vector is $\boldsymbol{x} = [\, 0.5, 0.5 \,]^T$.
- Each nonlinear equation → “curved” hypersurface (“flat” hyperplane for a linear equation) in $\mathbb{R}_n$
  - Linear Equations → 0,1 or ∞ many solutions
  - Nonlinear equations → any number of solutions.

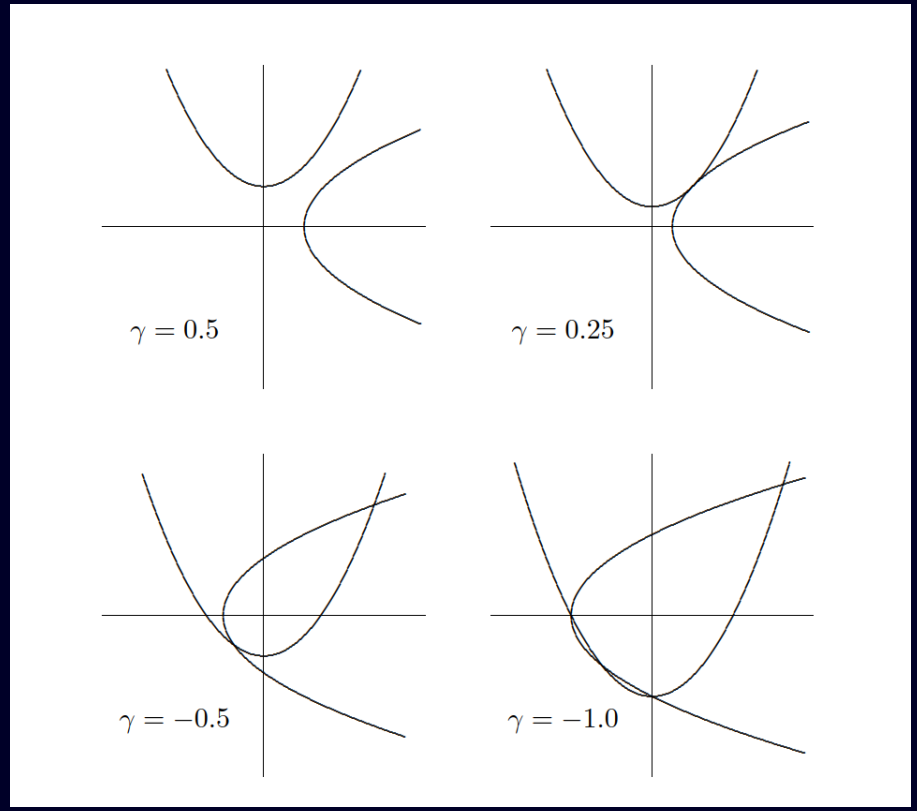
Exercise 1

Geometrical interpretation

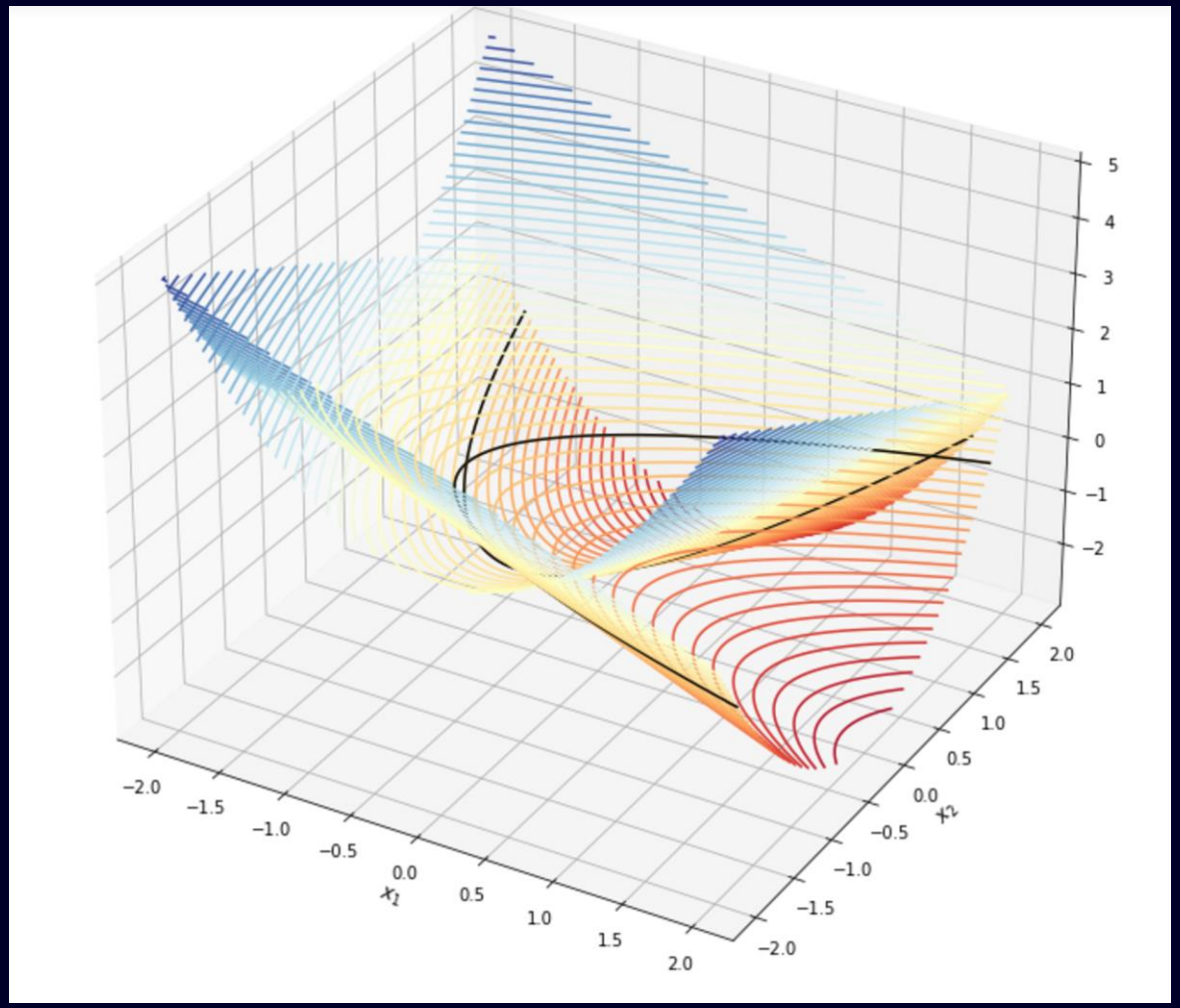
- Second order non-linear system of equations:

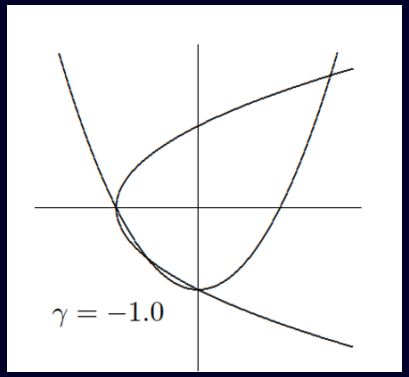
$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 - x_2 + \gamma \\ -x_1 + x_2^2 + \gamma \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Where γ is a parameter to be determined
- Each of the two component equations defines a **parabola**
- Any point where the two parabolas intersect is a **solution**
- Depending on the value for γ, this system can have either zero, one, two, or four solutions

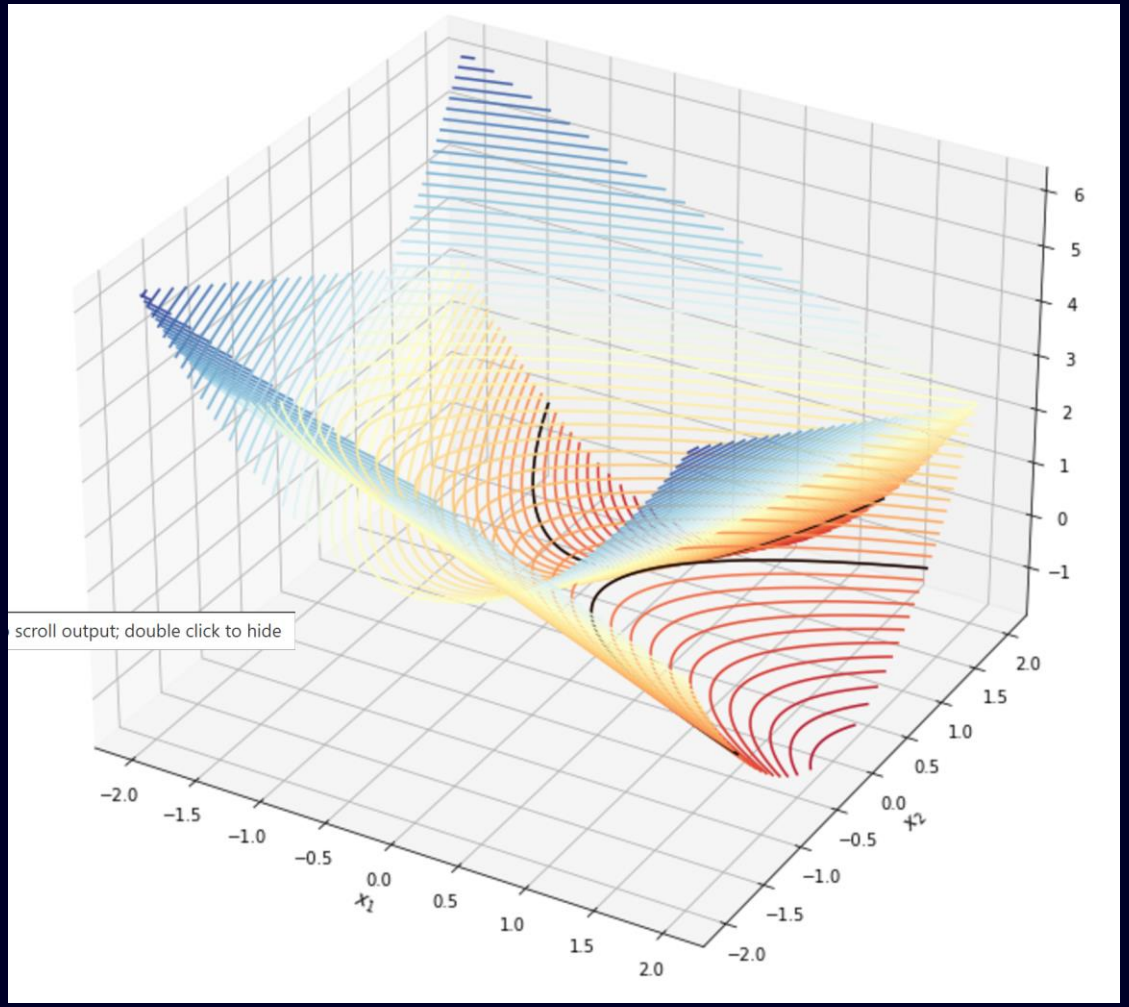


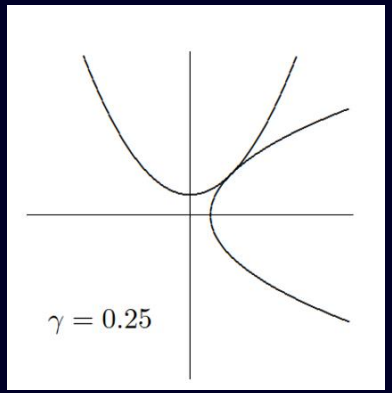
3D view: $\gamma = -1$





3D view: $\gamma = 0.25$





Existence and Uniqueness

Intermediate Value Theorem:

- if f is continuous on a closed interval $[a, b]$, and c lies between $f(a)$ and $f(b)$, then there is a value $x^* \in [a, b]$ such that $f(x^*) = c$.

- Thus, if $f(a)$ and $f(b)$ differ in sign, then by taking $c = 0$ in the theorem we can conclude that there must be a root within the interval $[a, b]$.
- Such an interval $[a, b]$ for which the sign of f differs at its endpoints is called **a bracket** for a solution of the one-dimensional nonlinear equation $f(x) = 0$.
- As we will see later, refining such a bracket plays an important part in some algorithms for finding a solution.
- Identifying such a bracket in the first place, however, is often a matter of trial and error.

Existence and Uniqueness

- Unlike linear equations, most nonlinear equations cannot be solved in a finite number of steps
- The Jacobian matrix plays an important role in the solvability

$$\{\boldsymbol{J}_f(\boldsymbol{x})\}_{ij} = \partial f_i(x)/\partial x_j$$

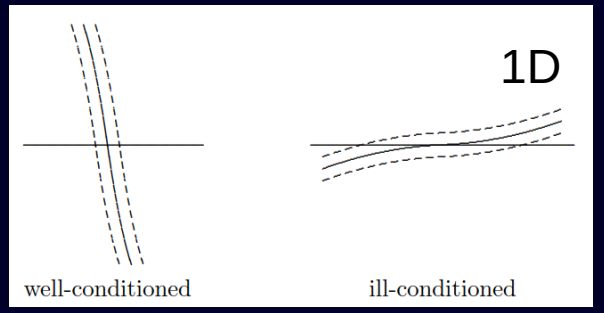
- If $\boldsymbol{J}_f(\boldsymbol{x})$ is nonsingular at a point x^*, then there is a neighborhood of $f(x^*)$ in which f is invertible, that is, the equation $f(x) = y$ has a solution for any y in that neighborhood of $f(x^*)$.

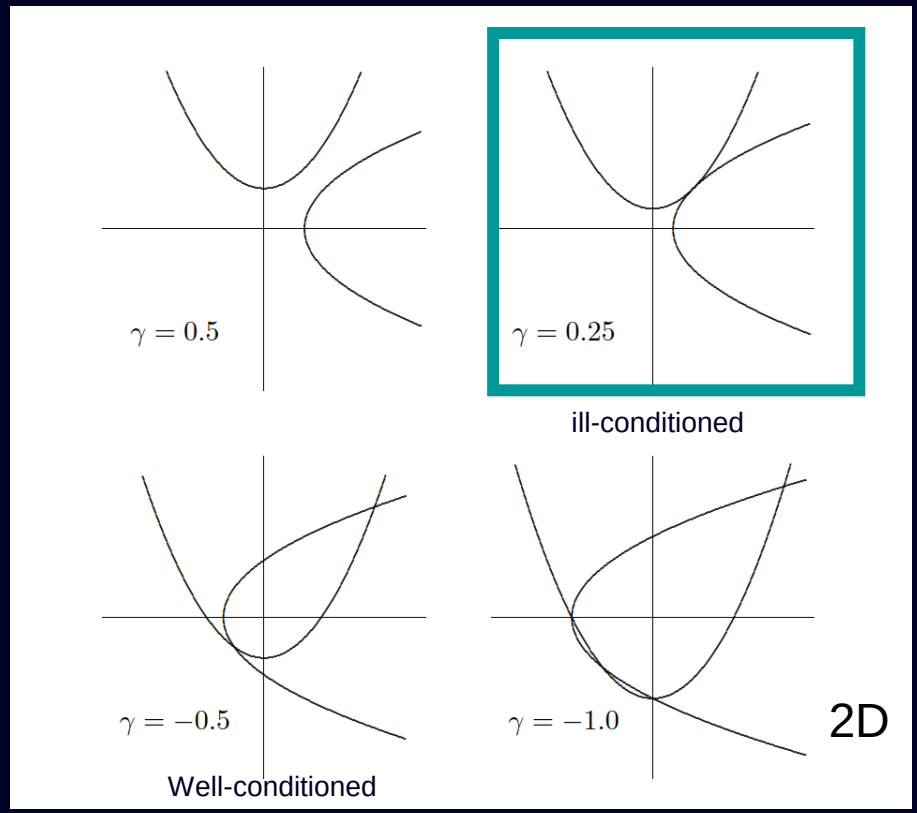
Conditioning and sensitivity

- The absolute condition number for root-finding problems in n-dimensions is $\{ \boldsymbol{J}_{\boldsymbol{k}}^{-1}(\boldsymbol{x}) \}$
- For the nonlinear system with $\gamma = 0.25$, the Jacobian matrix

$$\boldsymbol{J}_f(\boldsymbol{x}) = \begin{bmatrix} 2x_1 & -1 \\ -1 & 2x_2 \end{bmatrix}$$

- is singular at the unique solution $x^* = [0.5, 0.5]^T$.
  - slightly smaller $\gamma \Rightarrow$ 2 solutions
  - slightly larger $\gamma \Rightarrow$ no solution.





Convergence Rates and Stopping Criteria

- Unlike linear equations, most nonlinear equations cannot be solved in a finite number of steps
  - we resort to **iterative methods**, and use a stopping criterion (look for stagnation)
  - Total cost = cost per iteration × n_{iter} ⇒ Tradeoff between iteration cost and n_{iter}.

- **Convergence rate**
  - Error at iteration k: $e_k = x_k - x^*$, or interval length.
  - Convergence rate r:

$$\lim_{k\to\infty} \frac{\|\boldsymbol{e}_{k+1}\|}{\|\boldsymbol{e}_k\|^r} = C$$

1.	$10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, \ldots$	linear, with $C = 10^{-1}$
2.	$10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, \ldots$	linear, with $C = 10^{-2}$
3.	$10^{-2}, 10^{-3}, 10^{-5}, 10^{-8}, \ldots$	superlinear, but not quadratic
4.	$10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}, \ldots$	quadratic

- **Key Insight**
  - Faster convergence rates (higher r) lead to significantly reduced iterations for high accuracy.
  - Quadratic convergence means that potentially after 6 or 7 iterations → machine precision

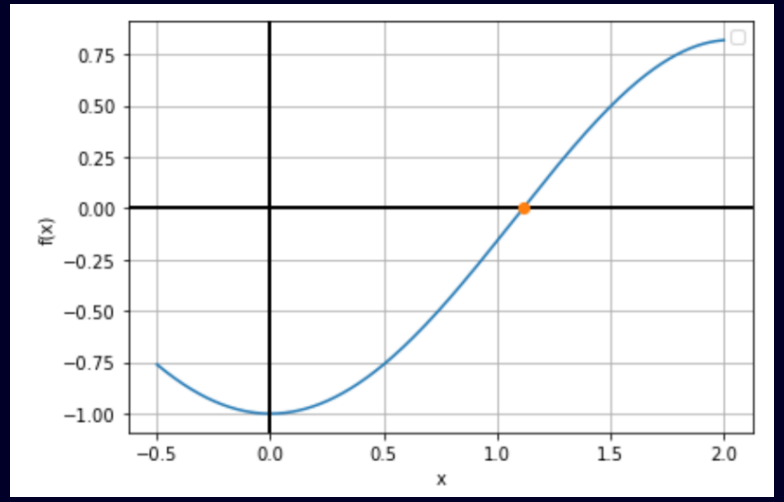
2. Standard algorithms (1D)

Preliminary analysis

- Let us consider the following 1D equation

$$f(x) = 0, \qquad x \in \mathbb{R}$$

- where f is a real-valued function with one parameter.
- We assume that this equation admits (at least) one root x^*, such that $f\,(x^*) \;=\; 0$.
- The idea is to build a sequence (x_n) that converges towards x^*.
- Hence, the term x_n of the sequence will be an approximation of x^*, the accuracy depending on the choice of n
- Main question?
  - How to build the sequence (x_n) ?

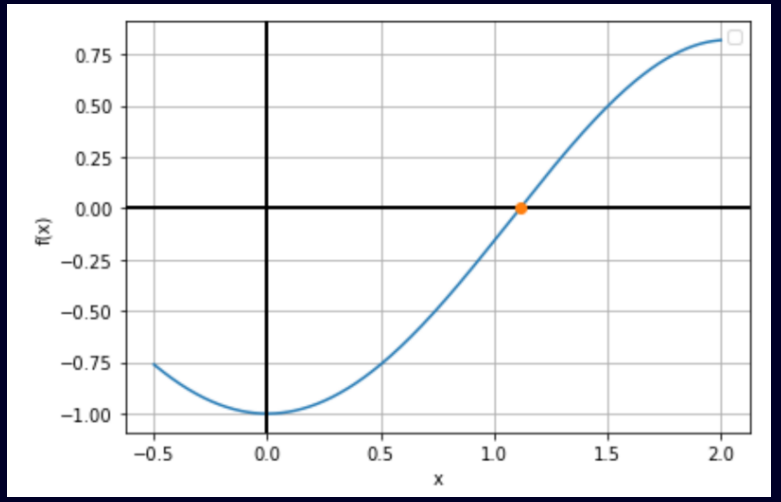


Preliminary analysis

- Let us consider the following 1D equation

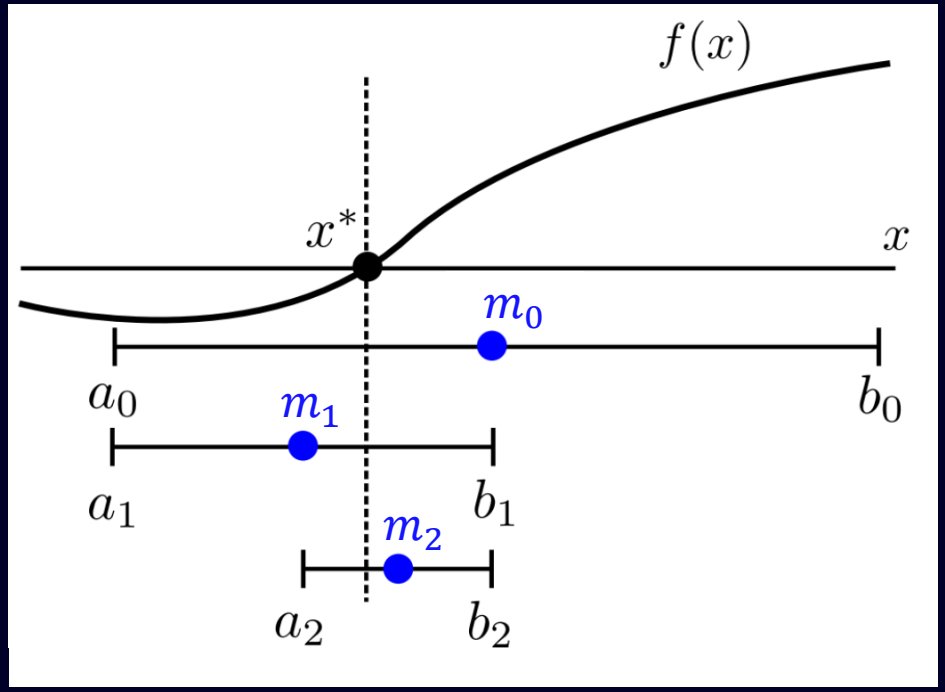
$$f(x) = 0, \qquad x \in \mathbb{R}$$

- where f is a real-valued function with one parameter.
- Before using a numerical method, it is better (if possible)
  - to check that the equation has at least one solution
  - to determine the number of roots
  - to localize the roots i.e. to determine some intervals $[a\,, b]$ in which the considered equation has one and only one solution



Bisection method

- **Principle of the method:**
  - We seek a very short interval $[a, b]$ where f changes of sign
  - To do so, we start with an initial interval that contains a root and we build a sequence of intervals such that
    - the root lies inside all the consecutive intervals (based on function sign switch)
    - the length of the intervals tends towards 0
- One gets a converging process for localizing the roots by subdividing



Bisection method

Pseudo-code

$a_0 = a,\ b_0 = b;$

forall *n from* 0 *to* N **do**

  $m := \frac{(a_n + b_n)}{2};$

  if $f(a)\, f(m) \leq 0$ **then**

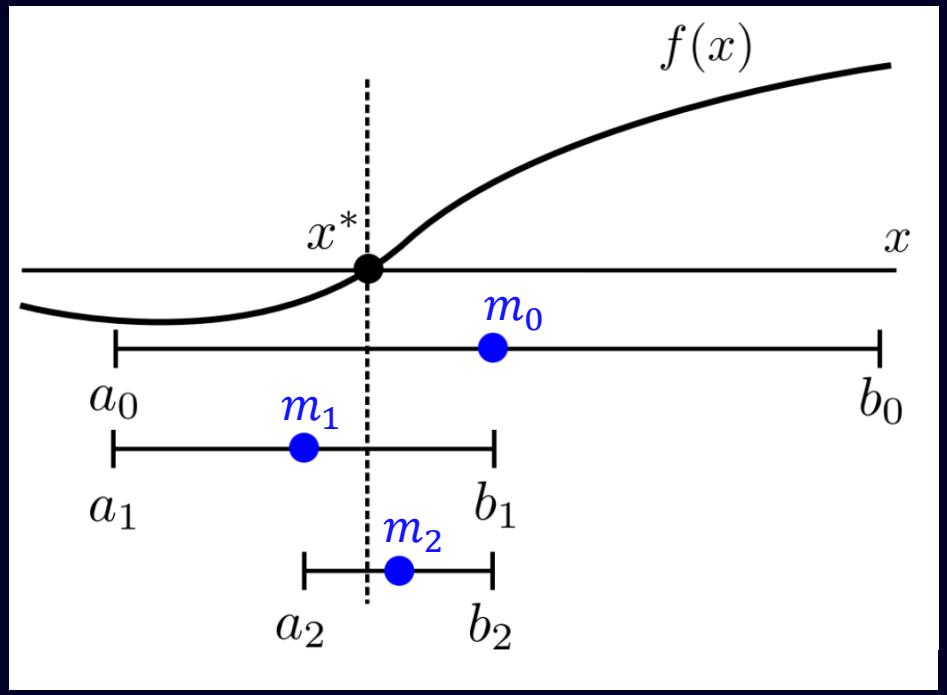
    $a_{n+1} := a_n,\ b_{n+1} := m;$

  else

    $a_{n+1} := m,\ b_{n+1} := b_n;$

  end

end



Exercise 2

Bisection method

- The bisection method makes no use of the magnitudes of the function values, only their signs
- As a result, bisection is **globally convergent** (provided interval contains a root) but does so rather slowly
- At each successive iteration, length of the interval containing the solution, and hence a bound on the possible error, is reduced by half → **linear convergence ($r = 1, C = 1/2$)**

$$\lim_{k\to\infty} \frac{\|\boldsymbol{e}_{k+1}\|}{\|\boldsymbol{e}_k\|^r} = C$$

- We have $a_k \leq x^* \leq b_k, \forall k \geq 0$, such that an upper bound for the error at iteration k:

$$|a_k - x^*| \leq |b_k - a_k| = \frac{|b-a|}{2^k}$$

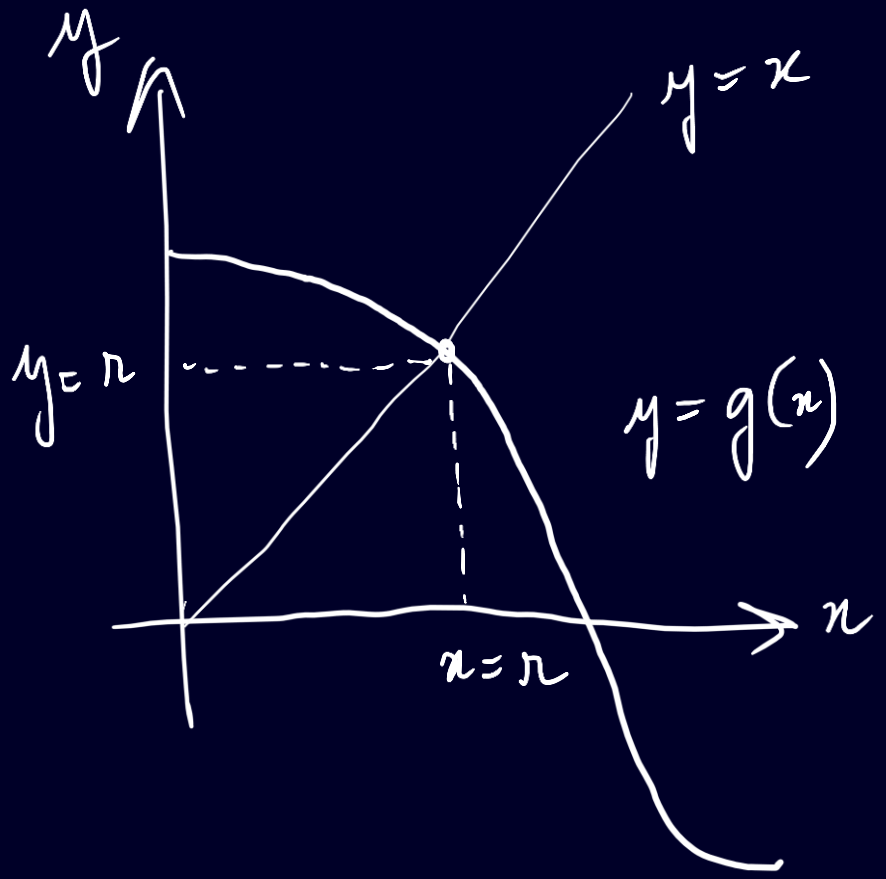
- Such that achieving an error tolerance of tol requires $\log_2|b-a|/tol$ iterations, **irrespective of f**

Fixed-point iteration

- If you type any number into a calculator and repeatedly hit the cosine button, the display eventually stops changing and gives you 0.739085
- Given a function $g\colon \mathbb{R} \to \mathbb{R}$, a value x such that

$$g(x) = x$$

- is called a **fixed point** of the function g
- We seek a point where g intersects the diagonal line
- Fixed-point problems often arise directly in practice
- But original problem are usually recast into their fixed-point form

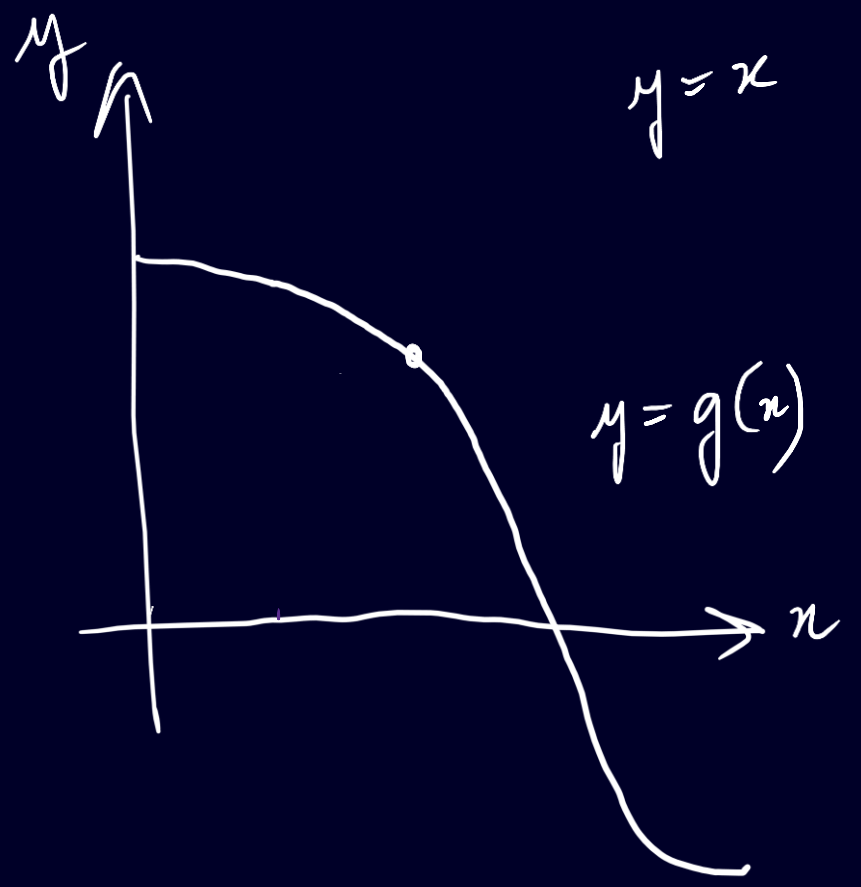


Fixed-point iteration

→ Just consists in repeatedly applying the fixed-form of non-linear equation $g(x)$ to an initial guess x_0

Pseudo-code

x_0 given;
forall *n from* 0 *to* ... **do**
  $x_{n+1} = g(x_n)$
end



Fixed point iteration: convergence

- If $x^*$is a fixed point, then for the error at the kth iteration we have

$$e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*)$$

- By the Mean Value Theorem, there is a point θ_k between x_k and x^* such that

$$g(x_k) - g(x^*) = g'(\theta_k)(x_k - x^*), \text{ so that } e_{k+1} = g'(\theta_k)e_k$$

- We do not know the value of θ_k, but if $|g'(x^*)| < 1$, then by starting the iterations close enough to x^*, we can be assured that there is a constant C such that $|g'(\theta_k)| \leq C < 1$, for $k = 0, 1, \ldots$

$$|e_k + 1| \leq C|e_k| \leq \cdots \leq C^k|e_0|$$

- $C < 1$ implies $C_k \to 0$, so $|e_k| \to 0$ and the sequence converges
- Hence provided $|g'(x^*)| < 1$ and we start close enough to the root, **convergence is linear**
- In the special case $\boldsymbol{g'(x^*) = 0}$, we can show that the **convergence is quadratic**

$$g(x_k) - g(x^*) = g''(\xi_k)(x_k - x^*)2/2$$

Fixed-point iteration

- Consider the function

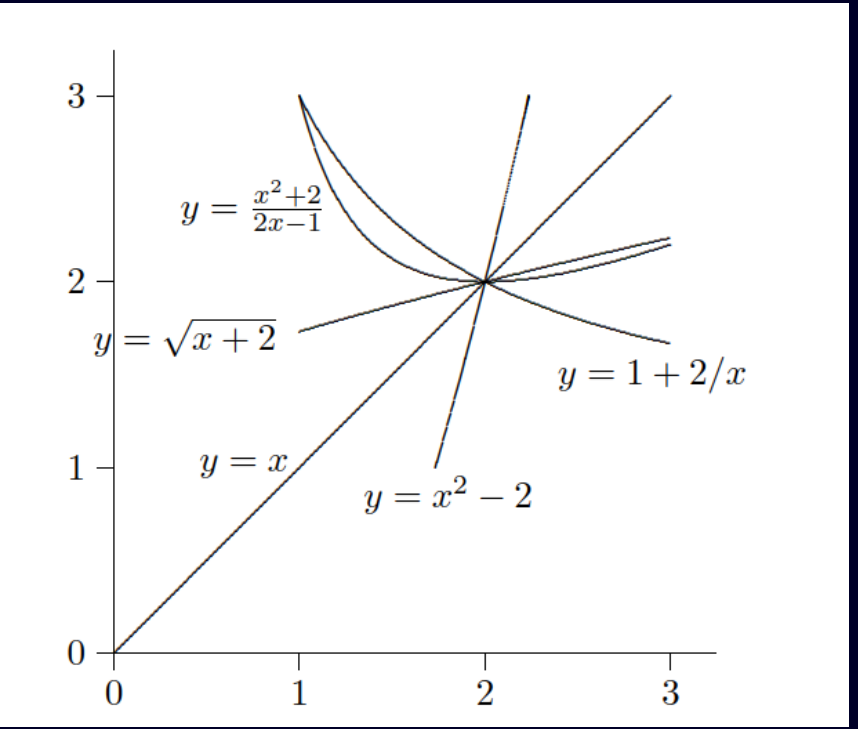
$$f(x) = x^2 - x - 2 = 0$$

- Which has roots $x^* = 2$ and $x^* = -1$.
- Equivalent fixed-point problems include

1. $g(x) = x^2 - 2$,
2. $g(x) = \sqrt{x+2}$,
3. $g(x) = 1 + 2/x$,
4. $g(x) = (x^2 + 2)/(2x - 1)$

- exhibiting divergence, slow convergence or rapid convergence

Exercise 3



If $|g'(x^*)| < 1$, then the iterative scheme is locally convergent

Fixed point iteration

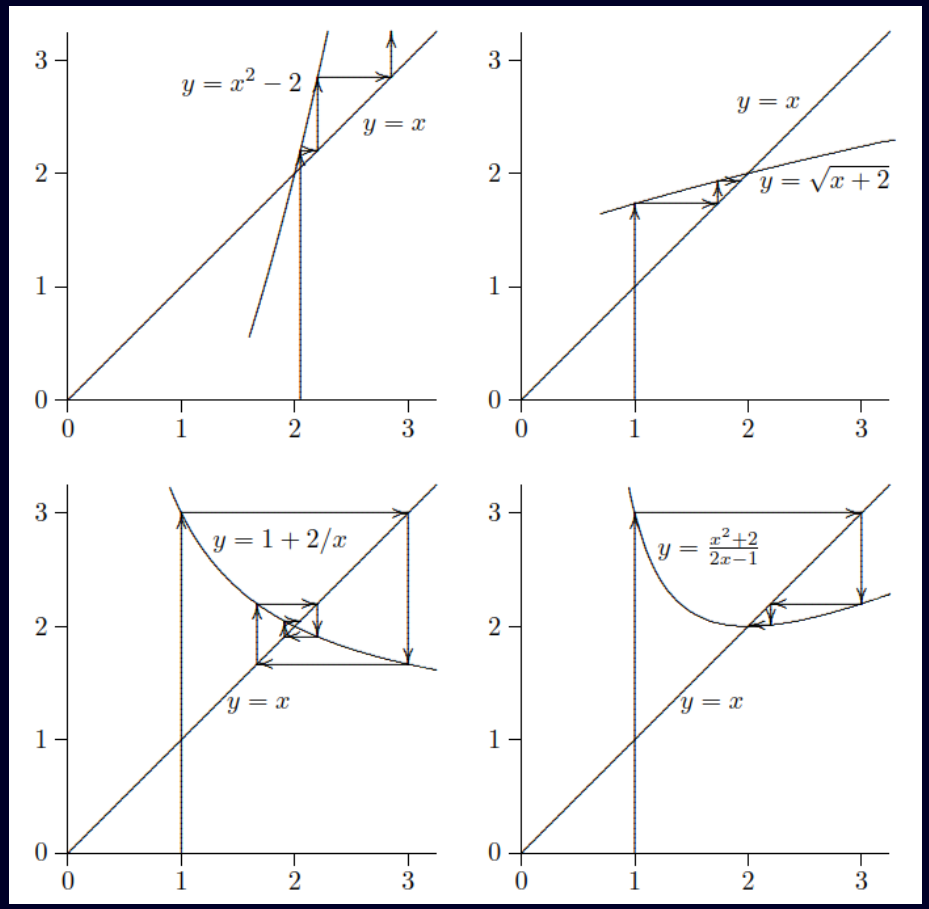
- Consider the function

$$f(x) = x^2 - x - 2 = 0$$

- Which has roots $x^* = 2$ and $x^* = -1$.
- Equivalent fixed-point problems include

1. $g(x) = x^2 - 2$,
2. $g(x) = \sqrt{x+2}$,
3. $g(x) = 1 + 2/x$,
4. $g(x) = (x^2+2)/(2x-1)$

- exhibiting divergence, slow convergence or rapid convergence



Fixed Point iteration

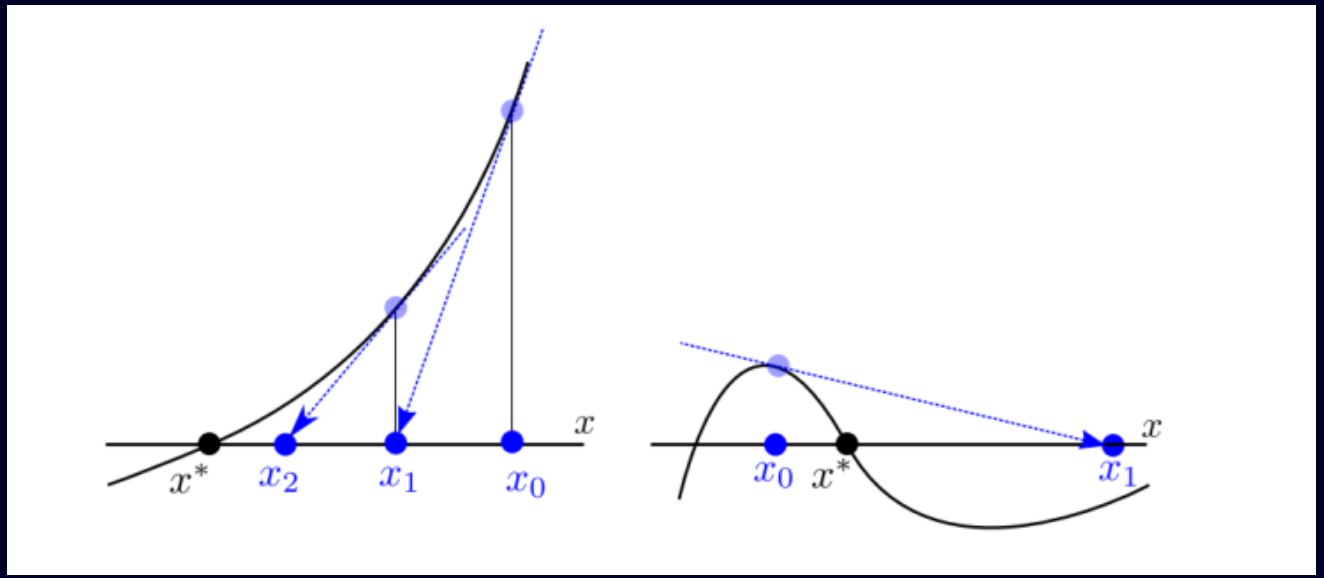
- For a given equation $f(x) = 0$, there may be many equivalent fixed-point problems $x = g(x)$ with different choices for the function g
- Not all fixed-point formulations are equally useful in deriving an iteration scheme for solving a given nonlinear equation.
- The resulting iteration schemes may differ not only in their convergence rates but also in **whether they converge at all**.
- If $|g'(x^*)| < 1$, then the iterative scheme is locally convergent

The Newton method

- if f is an affine function

$$f(x) = ax + b$$

- the root is $r = -b/a \ (a \neq 0)$
- the idea is to substitute f by an **affine approximation** → we can use its tangent.



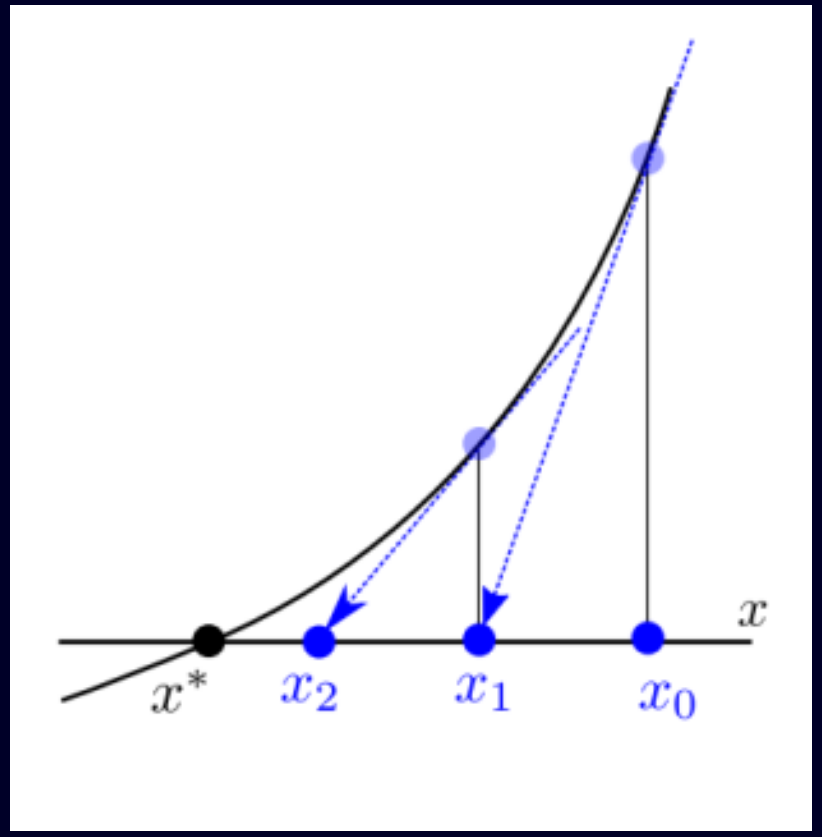
Newton: principle

- Assume that f is a function defined on an interval I, differentiable on I and such that it has a root r in I
- let x_0 be a point in I close enough to the root r , we then have

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \mathcal{O}(x - x_0)$$
$$= f_{x_0}(x) + \mathcal{O}(x - x0)$$

- the affine function f_{x_0} admits a root x_1 if and only if $f'(x0) \neq 0$, and in this case

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$



$$f(x + h) \approx f(x) + f'(x)h$$

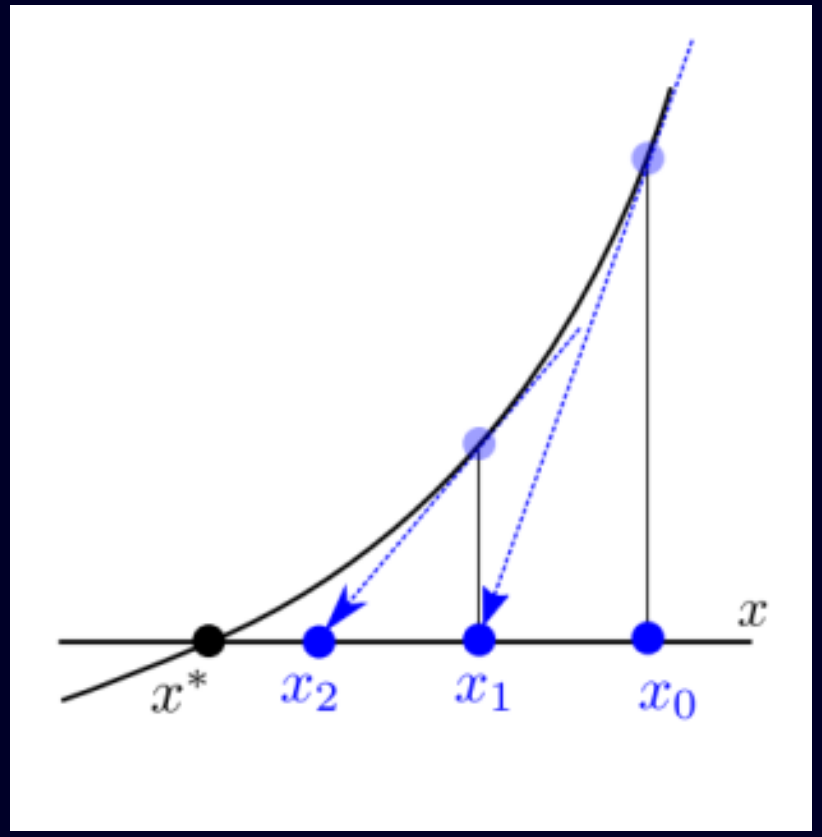
$$f(x + h) \approx 0 \rightarrow h = -f(x)/f'(x)$$

Newton: Pseudo code

→ Repeatedly subtracting function value divided by tangent to initial guess

Pseudo-code

x_0 given;
forall *n from* 0 *to* ... **do**
  $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
end



Newton: Convergence

- We can view Newton's method as a systematic way of transforming a nonlinear equation $f(x) = 0$ into a fixed-point problem $x = g(x)$, where

$$g(x) = x - f(x)/f'(x)$$

- To study the convergence of this scheme, we therefore determine the derivative

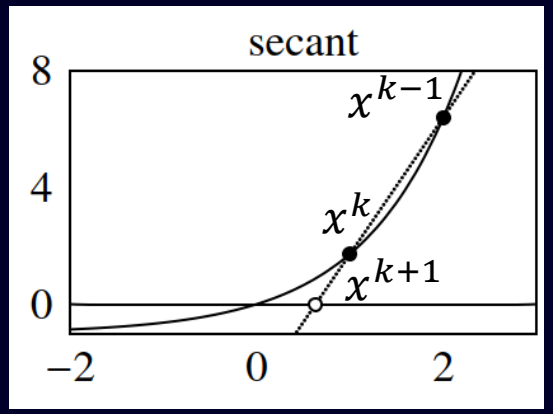
$$g'(x) = f(x)f''(x)/\left(f'(x)\right)^2$$

- If x^* is a simple root (i.e., $f(x^*) = 0$ and $f'(x*) \neq 0$), then $g'(x^*) = 0$. Thus, the **asymptotic convergence rate is quadratic** (for a simple root)
- However, this convergence result is **only local** Newton's method may fail if not close enough to the solution

Secant method

- Newton method ⇒ point of intersection of the line of slope $f'(x^k)$ passing through the point $(x^k, f(x^k))$
- Determining the derivative f' can be difficult analytically and numerically
- But we can use other slopes q^k that are suitable approximations to $f'(x^k)$

$$x^{k+1} = x^k - \frac{f(x^k)}{q^k}$$



- By taking the slope defined by two subsequent iterates

$$q^k = \frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}$$

- **Advantage**: Only one function evaluation per iteration
- **Drawback**: order of convergence is about 1.63 - a bit lower than Newton's method.

Exercise 4

Comparison between the algorithms

Bisection	Newton	Secant
• The method is globally convergent • Only one evaluation at each iteration • Convergence speed is linear and thereof slow	• Fast convergence when it converges • Not too sensitive to round-off errors if $f'(r)$ is not too small • Can diverge if the initial guess is not correctly chosen • Two evaluations at each iteration	• Relatively fast convergence when it occurs • Requires one evaluation at each iteration • Can diverge if the initial guess is not correctly calibrated

Example

Resolution of $x - 0.2 \sin x - 0.5 = 0$ with the four algorithms

	Bisection $x_{-1} = 0.5$ $x_0 = 1.0$	Secant $x_{-1} = 0.5$ $x_0 = 1.0$	Newton $x_0 = 1$	Fixed-point $x_0 = 1$ $x = 0.2 \sin x + 0.5$
1	0, 75	0, 5	0, 5	0, 50
2	0, 625	0, 61212248	0, 61629718	0, 595885
3	0, 5625	0, 61549349	0, 61546820	0, 612248
4	0, 59375	0, 61546816	0, 61546816	0, 614941
5	0, 609375			0, 61538219
6	0, 6171875			0, 61545412
7	0, 6132812			0, 61546587
8	0, 6152343			0, 61546779
9	0, 6162109			0, 61546810
10	0, 6157226			0, 61546815
11	0, 6154785			
12	0, 6153564			
13	0, 6154174			
14	0, 6154479			
15	0, 6154532			
16	0, 61547088			
17	0, 61546707			
18	0, 61546897			
19	0, 615468025			
20	0, 615468502			

Acceleration methods: Aitken's formula

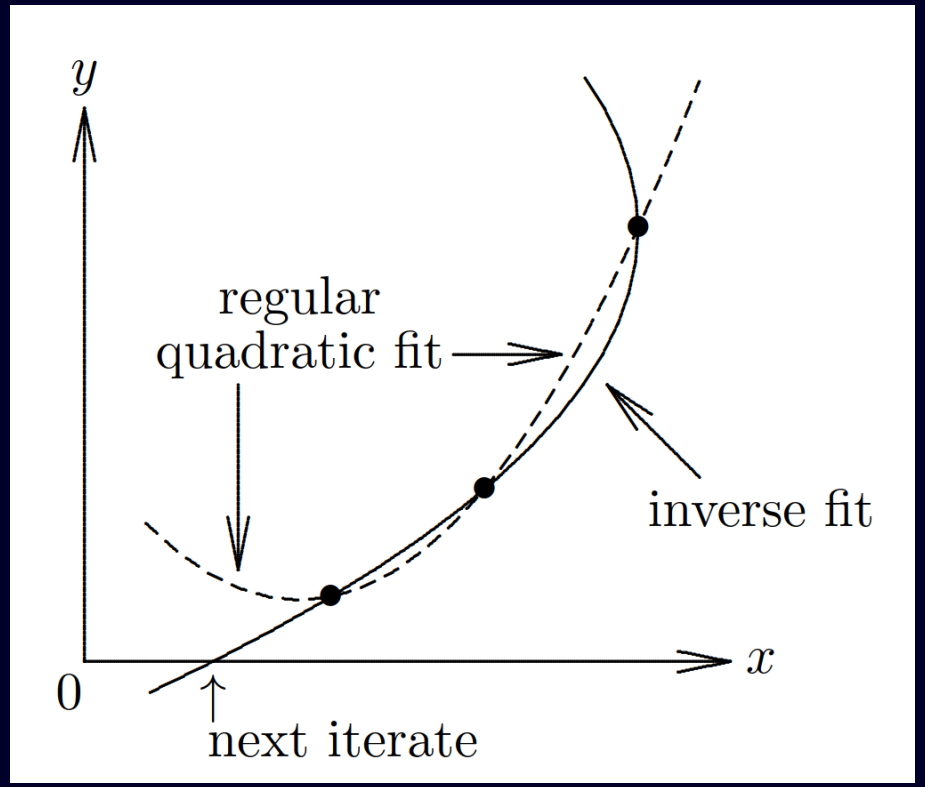
- Given a sequence $\{x_k\}$ from a fixed-point iteration $x_{\{k+1\}} = g(x_k)$
  - Aitken's accelerated sequence formula:

$$x_k^{\{(A)\}} = x_k - \left[\frac{\left(x_{\{k+1\}} - x_k\right)^2}{\left(x_{\{k+2\}} - 2x_{\{k+1\}} + x_k\right)}\right]$$

- **Algorithm Steps:**
  1. Perform standard fixed-point iterations to generate $x_k, x_{\{k+1\}}, x_{\{k+2\}}$.
  2. Apply Aitken's formula to compute $x_k^{\{(A)\}}$ & use as updated approximation for subsequent iterations.
- **Benefits:**
  - Faster convergence when the sequence is close to a fixed point.
  - Computationally inexpensive to implement.
- **Limitations:**
  - Requires three consecutive iterates $\left(x_k, x_{\{k+1\}}, x_{\{k+2\}}\right)$
  - Sensitive to numerical errors if denominator ≈ 0.

Inverse quadratic interpolation

- Given three points $(x1, f(x1)), (x2, f(x2)), (x3, f(x3))$, construct a quadratic function that passes through these points.
- Best is to use **inverse interpolation**, in which one
  - fits the values x_k as a function of the values $y_k = f(x_k)$
  - by a polynomial $p(y)$
  - next approximate solution is simply $p(0)$
- **Advantages**:
  - Faster convergence than linear interpolation (e.g., Secant method).
  - No need for derivative evaluations.
- **Limitations**:
  - Requires three distinct points
  - Sensitive to numerical instability if points are close together



Safeguarded methods

- **Dilemma:**
  - Rapidly convergent methods (e.g., Newton's, Secant) are unsafe unless started close to the solution.
  - Safe methods (e.g., Bisection) are slow and costly.
- **Solution**:
  - Use hybrid methods that combine features of both.
  - **Example**: Use a rapidly convergent method but maintain a bracket around the solution.
    - If the next approximate solution falls outside the bracket, fall back to a safe method (e.g., Bisection).
    - Try the rapid method again on a smaller interval. Fast convergence usually prevails.
- **Popular Implementation**:
  - **Brent's method**: Combines safety of Bisection with faster convergence of inverse quadratic interpolation.
    - https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.brentq.html
  - Avoids Newton's method, so derivatives are not required.
  - Addresses potential pitfalls in floating-point arithmetic (e.g., underflow, overflows, tight tolerances).

3. Systems of non linear equations

Problem description

- Let us consider the equation

$$F(X) = 0$$

- Where $F: \mathbb{R}_n \to \mathbb{R}_n$ or in terms of scalar equations

$$\begin{cases} f_1(x_1, x_2, \ldots, x_N) & = 0 \\ f_2(x_1, x_2, \ldots, x_N) & = 0 \\ \vdots & \vdots \\ f_N(x_1, x_2, \ldots, x_N) & = 0 \end{cases}$$

Newton-Raphson method

- the Newton-Raphson method is a generalization to higher-dimensional problems of the one-dimensional Newton method

$$x_{n+1} = x_n - (f'(x_n))^{-1} f(x_n)$$

- it involves the Jacobian matrix of F:

$$F'(X_n) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{pmatrix}$$

all the derivatives being evaluated at point X_n.
The Newton-Raphson method formally writes down

$$X_{n+1} = X_n - [F'(X_n)]^{-1} F(X_n)$$

Taylor Series for vector functions

Theorem

Let $X = (x_1, x_2, \ldots, x_n)^T$, $F = (f_1, f_2, \ldots, f_m)^T$, and assume that $F(X)$ has bounded derivatives up to order at least two. Then for a direction vector $P = (p_1, p_2, \ldots, p_n)^T$, the Taylor expansion for each function f_i in each coordinate x_j yields

$$F(X + P) = F(X) + F'(X)P + \mathcal{O}\left(\|P\|^2\right),$$

where $F'(X)$ is the Jacobian matrix of first derivatives of F at X. Thus we have

$$f_i(X + P) = f_i(X) + \sum_{i=1}^{n} \frac{\partial f_i}{\partial x_j} p_j + \mathcal{O}\left(\|P\|^2\right), \quad i = 1, \ldots, m$$

For a small $P = R - X_n$, we have $0 = F(X_n + P) \approx F(X_n) + F'(X_n)(R - X_n)$.
We then define δ_n such as $F(X_n) + F'(X_n)\delta_n = 0$

The Newton-Raphson method

- In a practical computation, we do not explicitly compute the inverse of the Jacobian matrix which would be too expensive
- We prefer to write the algorithm under the following form
- For X_0 given

Pseudo-code

forall *n from* 0 *to* . . . **do**
  Solve the linear system $F'(X_n)\delta_n = -F(X_n)$;
  $X_{n+1} = X_n + \delta_n$;
end

Remarks

- The choice of the initial guess is crucial and the risk that the algorithm diverges truly exists
- The convergence is second-order and therefore is really fast (when it converges!)
- The Newton Raphson method is expensive at each iteration, one must
  - Evaluate $N^2 + N$ functions (N^2 Jacobian matrix and N coordinate functions)
  - Solve the linear system of size $N \times N$ (with a dense matrix)

Exercise 6 and 7

3. Summary

Summary

- **We have seen a few methods to find roots of 1D non-linear equations.**
  - the fixed-point theory is a fundamental concept to develop algorithms,
  - they are methods to accelerate convergence (Aitken-acceleration)
  - built-in methods combine the bisection, Newton and secant methods
- **In two or more dimensions the situation is more complicated**
  - Newton method can still be used, but is very costly
  - Cheaper methods can be devised by approximating the Jacobian
  - Root-finding is strongly linked to optimization